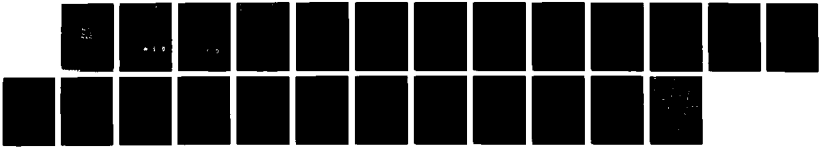
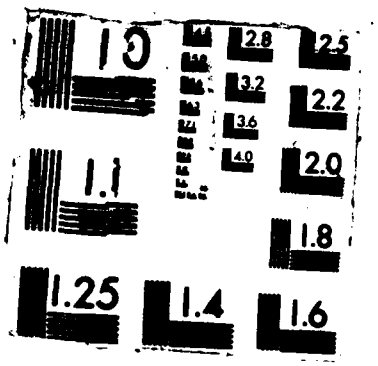
AD-A183 373 A CLASS OF OPTIMAL DECENTRALIZED COMMIT PROTOCOLS(U) 1/1
MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE
A K AGRAWALA ET AL. JUL 87 CS-TR-1873 N00014-87-K-0124
UNCLASSIFIED F/G 25/5 NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UMIACS-TR-87-31
CS-TR-1873

July, 1987

A Class of Optimal Decentralized Commit Protocols

Shyan-Ming Yuan
Department of Computer Science

A. K. Agrawala
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

COMPUTER SCIENCE TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

(12)

UMIACS-TR-87-31
CS-TR-1873

July, 1987

A Class of Optimal Decentralized Commit Protocols

Shyan-Ming Yuan
Department of Computer Science

A. K. Agrawala
Department of Computer Science and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

ABSTRACT

This paper studies the message complexity of decentralized commit protocols. It shows that $\Theta(kNN^{1/k})$ messages are necessary only if k rounds of message interchanges are allowed. It also shows that $\Theta(N \ln N)$ is a message lower bound for any decentralized commit protocol. Finally, a class of decentralized commit protocols are proposed which need $\Theta(kNN^{1/k})$ messages and use k rounds of message interchanges. If we let $k = \ln N$ then we can get a decentralized commit protocol which needs $\Theta(N \ln N)$ messages only.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA183373

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION / AVAILABILITY OF REPORT approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CS-TR-1873; UMIACS-TR-87-31	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION University of Maryland 6b OFFICE SYMBOL (If applicable) N/A	7a NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) Department of Computer Science University of Maryland College Park, MD 20742	7b ADDRESS (City, State, and ZIP Code) 800 North Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-87-K-0124
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
A Class of Optimal Decentralized Commit Protocols

12 PERSONAL AUTHOR(S)
Shyan-Ming Yuan, Ashok K. Agrawala

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM _____ TO _____	July 1987	21

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper studies the message complexity of decentralized commit protocols. It shows that $\Theta(kNN^{1/k})$ messages are necessary only if k rounds of message interchanges are allowed. It also shows that $\Theta(N\ln N)$ is a message lower bound for any decentralized commit protocol. Finally, a class of decentralized commit protocols are proposed which need $\Theta(kNN^{1/k})$ messages and use k rounds of message interchanges. If we let $k=\ln N$ then we can get a decentralized commit protocol which needs $\Theta(N\ln N)$ messages only.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code) / 22c OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

1 Introduction

Decentralized commit protocols are characterized by successive rounds of message interchanges. Protocols in [1, 2] which require a site to communicate with every other site at each step need $N(N-1)$ and $2N(N-1)$ messages for each round, respectively. N is the number of sites in the system. Protocols in [4] need $k^2NN^{1/k}$ and $2k^2NN^{1/k}$ messages for blocking and nonblocking protocols, respectively. Where k is a parameter dependent on the number of rounds of message exchange. In this paper, we define a logical communication structure develop a family of blocking and nonblocking commit protocols. The protocols need only $kNN^{1/k}$ and $2kNN^{1/k}$ messages, where k is the same integer parameter of [4]. This communication scheme can also be used for decentralized consensus protocols [5] with the same message complexity.

In section 2, we describe the formal model for our commit protocols. In section 3, we describe the decentralized versions of two-phase blocking and nonblocking protocols. In section 4, we show that the message complexity of decentralized commit protocol. We present a class of message optimal commit protocols in section 5. We also extend this idea to some decentralized consensus protocols in section 6.

2 Formal Model for Commit Protocols

Transaction execution at each site is modeled by a finite state automation (FSA) [3]. The network serves as a common I/O medium for all the FSA's. A local state transition at site i involves the reading of a nonempty string of messages addressed to site i, writing a string of messages, and moving to the next local state. State transitions at one site are asynchronous with respect to transitions at other sites. The final states of the FSA's are partitioned into two sets: the abort and the commit states. A site cannot make transitions to nonabort states from abort states. Similarly, a site cannot make transitions from commit states to noncommit states.

A local state is called commitable if occupancy of that state by any site implies that all sites have agreed to commit the transaction. All other states are said to be noncommitable. The global state of a distributed transaction can be represented by a global state vector consisting of the local states of all FSAs and the outstanding messages in the network. The set of all global states reachable from the initial global state can be represented by a global reachable graph. Using this global reachable graph, it is possible to find out the set of local states that may concurrently be occupied by other FSAs when a FSA at site i is in some known state Si. This set of states that can concurrently be occupied by FSAs at other sites is called the concurrency set of Si.

3 Blocking and Nonblocking Commit Protocols

The FSA for the decentralized two phase commit protocol is shown in figure 1. We assume that each site has already received a transaction in the initial state (state q). If a site decides to commit the transaction, it sends 'yes' messages to all the other $N-1$ sites and moves to state w. If a site decides to abort the transaction it sends 'no' messages and moves to state a. In state w, if 'yes' messages are received from all other sites the transaction is commited (state c) otherwise the transaction is aborted, since the concurrency set of state w contains c and a. Skeen [2] has shown that a protocol is nonblocking if and only if:

1. there exists no local state whose concurrency set contains both commit and abort states.

2. there exists no noncommittable state whose concurrency set contains a commit state.

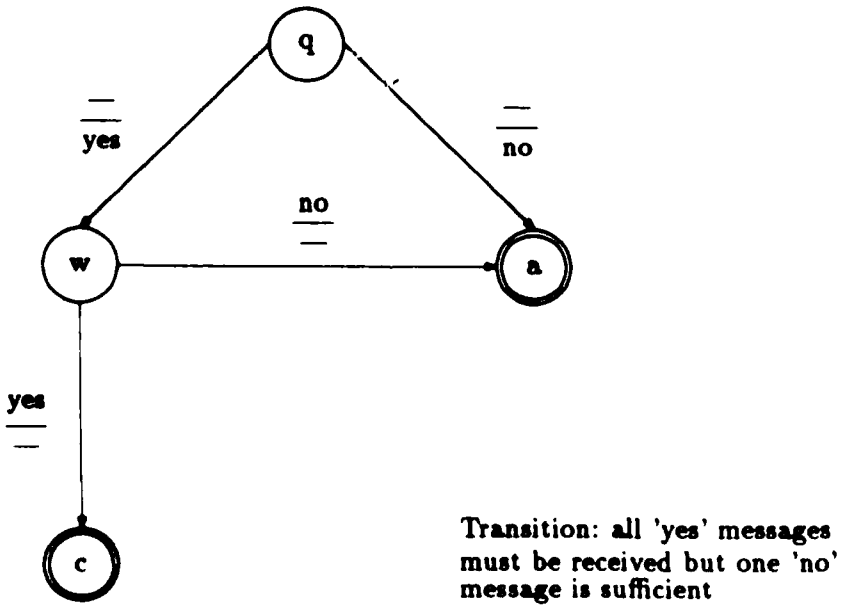


Figure 1: Blocking Protocol

Therefore the protocol is a blocking protocol. Figure 2 illustrates the decentralized nonblocking commit protocol described in [2]. This protocol has a buffer state p and an extra round of messages to make it conform the requirements of Skeen's nonblocking theorem.

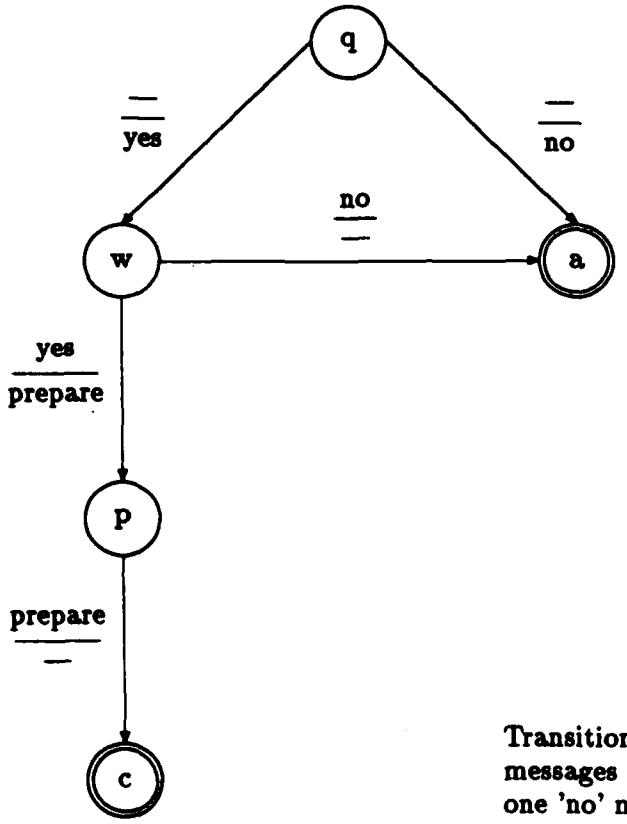


Figure 2: Nonblocking Protocol

4 The Message Complexity of Decentralized Commit Protocols

THEOREM 4.1 *For all decentralized commit protocols, if there are only k rounds of message interchanges allowed then* $\Theta(kNN^{1/k})$ *messages are necessary.*

PROOF: Assume each round of message interchange, each site can only write p messages out. Therefore, after round 1, there are at most $(p+1)$ sites having the knowledge about the local decision of site 1. After round 2, there are at most $(p+1)^2$ sites having the knowledge about the local decision of site 1. Therefore, after round k, there are at most $(p+1)^k$ sites having the knowledge about the local decision of site 1. Since only k rounds of message interchanges are allowed, $(p+1)^k$ should be at least N. Otherwise, these N sites can not enter a common final state. Because during each round, each site writes p messages out, total messages generated are Npk.

$$\begin{aligned} (p+1)^k &\geq N \\ (P+1) &\geq N^{1/k} \\ p &\geq N^{1/k} - 1 \end{aligned} \tag{1}$$

$$\begin{aligned} Npk &\geq kN(N^{1/k} - 1) \\ Npk &= \Theta(kNN^{1/k}) \end{aligned} \tag{2}$$

□

THEOREM 4.2 *For all decentralized commit protocols,* $\Theta(N \ln N)$ *messages are necessary.*

PROOF: All decentralized commit protocols, which need k rounds of message interchanges, require $\Theta(kNN^{1/k})$ messages. Therefore, minimizing $kNN^{1/k}$ will get a lower bound for all decentralized commit protocols.

$$\begin{aligned} \frac{d(kNN^{1/k})}{dk} &= 0 \\ NN^{1/k} + kN(\frac{-\ln N}{k^2})N^{1/k} &= 0 \end{aligned}$$

$$NN^{1/k} = NN^{1/k}(\frac{\ln N}{k})$$

$$k = \ln N \tag{3}$$

$$\min(kNN^{1/k}) = N^{1/(\ln N)}N \ln N$$

$$= eN \ln N \tag{4}$$

$$\min(\Theta(kNN^{1/k})) = \Theta(N \ln N) \tag{5}$$

So, $\Theta(N \ln N)$ messages are necessary for any decentralized commit protocol. □

5 Families of Commit Protocols

In this section two families of decentralized commit protocols are presented. Assume that N and k are such that $N^{1/k}$ is an integer (later we consider the case when it is not an integer). The N sites can be treated as N positions of a k-dimensional array such that a site x can be numbered by a k-tuple $(X_1, X_2, \ldots, X_k)$, where $0 \leq X_1, X_2, \ldots, X_k \leq (N^{1/k} - 1)$ and

$$\forall x, 0 \leq x \leq N-1, x = \sum_{j=1}^{k} X_j * N^{(k-j)/k}$$

5.1 Protocol Schema 1

The FSA for schema 1 is shown in figure 3. The actions in each state are listed below. For all site x, $x = 0, 1, \ldots, N-1$, x can be numbered as k-tuple $(X_1, X_2, \ldots, X_k)$.

1. State q: On receiving a transaction, unilaterally decide to commit or abort the transaction. If a 'no' message is received, decide to abort the transaction. If an abort decision is made, send 'no' messages and move to state a_1. If a commit decision is made, send 'yes^1' messages and move to state w_1. All messages are sent to sites numbered as k-tuple $(*, X_2, \ldots, X_k)$, where '*' means any number between 0 and $N^{1/k} - 1$.

2. State w_i, $1 \leq i \leq (k-1)$: If all 'yes^i' messages are received from sites numbered as k-tuple $(X_1, \ldots, X_{i-1}, *, X_{i+1}, \ldots, X_k)$. Then, send '$yes^{i+1}$' messages and move to state w_{i+1}. If a 'no' message is received , send 'no' messages and move to state a_{i+1}. All messages are sent to sites numbered as k-tuple $(X_1, \ldots, X_i, *, X_{i+2}, \ldots, X_k)$.

3. State w_k: If all 'yes^k' messages are received from sites numbered as $(X_1, \ldots, X_{k-1}, *)$, commit the transaction. If a 'no' message is received, abort the transaction.

4. State c: Commit state.

5. State a_i, $1 \leq i \leq (k-1)$: Send 'no' message to sites $(X_1, \ldots, X_i, *, X_{i+2}, \ldots, X_k)$ and move to state a_{i+1}.

6. State a_k: Abort state.

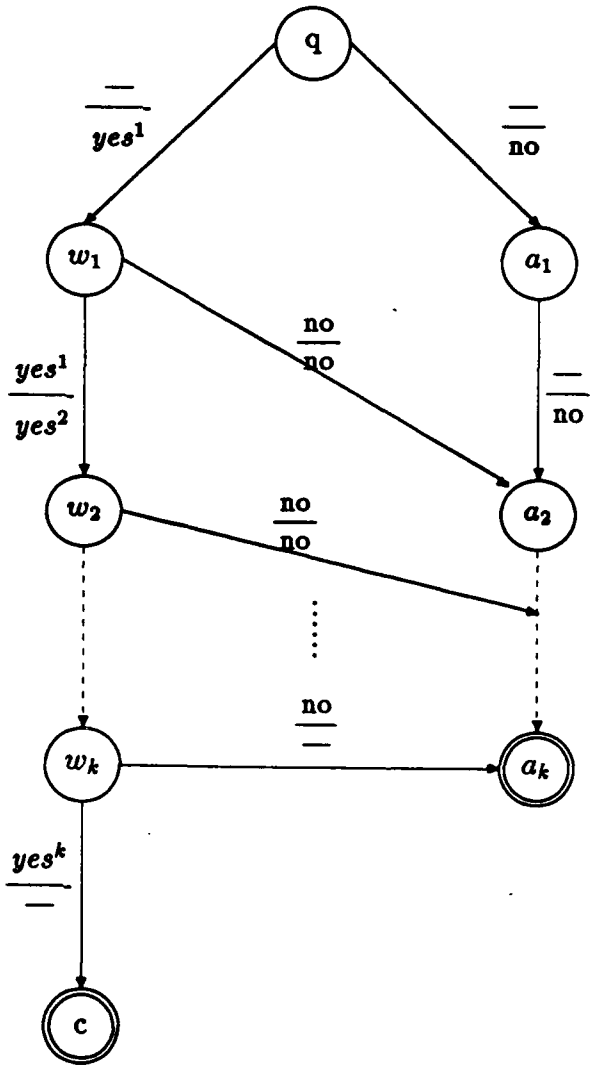


Figure 3: Protocol Schema 1

The following theorems establish the properties of protocol schema 1.

LEMMA 5.1 *If a site is in one of a states, it will eventually abort.*

PROOF: Since the only transition in a states is to send 'no' message and move to the next a state. Hence, if a site is in one of a states, it will eventually move to state a_k (abort state). □

THEOREM 5.1 *If a site decides to abort, all sites will eventually abort.*

PROOF: Assume that site x decides to abort in State q, where x can be numbered as k-tuple $(X_1, X_2, \ldots, X_k)$. Then all sites will abort after kth round of message exchange. We will prove this by induction.

INDUCTIVE HYPOTHESIS: *If site x decides to abort. After ith round of message exchange, where* $1 \leq i \leq k$, *all sites numbered as* $(\underbrace{*, \ldots, *}_{i-bits}, X_{i+1}, \ldots, X_k)$ *have received 'no' message and will abort. (The '*' in any position indicates that the element in that position can take any value between 0 and* $N^{1/k} - 1$.*)*

1. Base case: When i is equal to 1. Since site x decides to abort in State q. It sends 'no' messages to sites which are numbered as $(*, X_2, \ldots, X_k)$. Therefore, after the 1st round of message exchange all sites numbered as $(*, X_2, \ldots, X_k)$ have received 'no' messages and will move to state a_2. By Lemma 4.1, They will abort eventually.

2. Inductive case: Assume that the hypothesis is true for $i = l - 1$, where $l \leq k$. After $(l-1)$th round of message exchange, all sites numbered as $(\underbrace{*, \ldots, *}_{(l-1)bits}, X_l, \ldots, X_k)$ have received 'no' message. Therefore, they will move to a states and send 'no' message to all sites numbered as $(\underbrace{*, \ldots, *}_{l-bits}, X_{l+1}, \ldots, X_k)$. Such that after lth round of message exchange, all sites numbered as $(\underbrace{*, \ldots, *}_{l-bits}, X_{l+1}, \ldots, X_k)$ have received 'no' message. By lemma 4.1, They will abort eventually. So the inductive hypothesis is true for all i, where $1 \leq i \leq k$.

Therefore, If a site x decides to abort, after kth round of message exchange all sites numbered as $(\underbrace{*,\ldots,*}_{k-bits})$ will abort. So if a site decides to abort, all sites will eventually abort. □

THEOREM 5.2 *If no site decides to abort the transaction, all sites will eventually commit.*

PROOF: If no sites aborts the transaction, all sites will send 'yes^1' and move to state w_1. Since all messages are eventually delivered, each site will receive all 'yes^1' messages needed, then send 'yes^2' message and move to state w_2. Similarly, for all other 'yes' messages. Finally, all 'yes^k' messages will be delivered causing sites to move to the commit state. □

THEOREM 5.3 *Protocol 1 is a blocking protocol.*

PROOF: It is obvious that the concurrency set of state w_k contains both abort and commit states. Hence, protocol 1 violates the nonblocking conditions states in section 3. □

5.2 Protocol Schema 2

The FSA for schema 2 is shown in figure 4. The actions in each state are listed below. For all site x, $x = 0, 1, \ldots, N-1$, x can be numbered as k-tuple $(X_1, X_2, \ldots, X_k)$.

1. State q: On receiving a transaction, unilaterally decide to commit or abort the transaction. If a 'no' message is received, decide to abort the transaction. If an abort decision is made, send 'no' messages and move to state a_1. If a commit decision is made, send 'yes^1' messages and move to state w_1. All messages are sent to sites numbered as k-tuple $(*, X_2, \ldots, X_k)$, where '$*$' means any number between 0 and $N^{1/k} - 1$.

2. State w_i, $1 \leq i \leq (k-1)$: If all 'yes^i' messages are received from sites numbered as k-tuple $(X_1, \ldots, X_{i-1}, *, X_{i+1}, \ldots, X_k)$. Then, send '$yes^{i+1}$' messages and move to state w_{i+1}. If a 'no' message is received, send 'no' messages and move to state a_{i+1}. All messages are sent to sites numbered as k-tuple $(X_1, \ldots, X_i, *, X_{i+2}, \ldots, X_k)$.

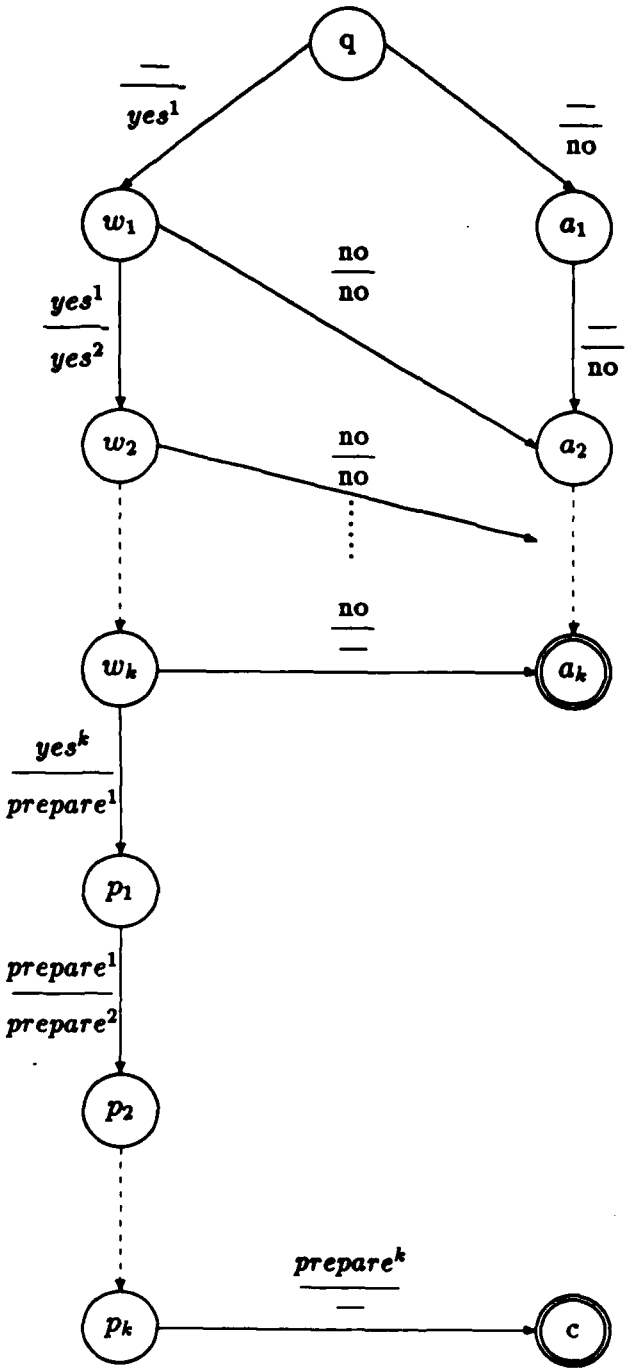


Figure 4: Protocol Schema 2

3. State w_k: If all 'yes^k' messages are received from sites $(X_1, \ldots, X_{k-1}, *)$, send '$prepare^1$' messages to sites numbered as k-tuple $(*, X_2, \ldots, X_k)$ and move to state p_1. If a 'no' message is received, move to state a_k (abort state).

4. State p_i, $1 \leq i \leq (k-1)$: If all '$prepare^i$' are received from sites numbered as $(X_1, \ldots, X_{i-1}, *, X_{i+1}, \ldots, X_k)$. Then, send '$prepare^{i+1}$' messages to sites numbered as $(X_1, \ldots, X_i, *, X_{i+2}, \ldots, X_k)$ and move to state p_{i+1}.

5. State p_k: If all '$prepare^k$' are received from sites $(X_1, \ldots, X_{k-1}, *)$, commit the transaction.

6. State c: Commit state.

7. State a_i, $1 \leq i \leq (k-1)$: Send 'no' messages to sites which are numbered as k-tuple $(X_1, \ldots, X_i, *, X_{i+2}, \ldots, X_k)$ and move to state a_{i+1}.

8. State a_k: Abort state.

The following theorems establish the properties of protocol schema 2.

THEOREM 5.4 *If a site decides to abort, all sites will eventually abort.*

PROOF: Similar to proof of theorem 4.1.

THEOREM 5.5 *If no site aborts the transaction, all sites will eventually commit.*

PROOF: Similar to proof of theorem 4.2.

LEMMA 5.2 *If a site is in state c, all sites have sent '$prepare^1$' messages.*

PROOF: We would like to show it by contradiction. Assume that site x is in state c and site y has not sent '$prepare^1$' yet. Where x is numbered as $(X_1, \ldots, X_k)$ and y is numbered as $(Y_1, \ldots, Y_k)$. Since y has not sent '$prepare^1$', all sites numbered as $(*, Y_2, \ldots, Y_k)$ have not received all needed '$prepare^1$' messages. They should have not sent '$prepare^2$' yet. Therefore, all sites numbered as $(*, *, Y_3, \ldots, Y_k)$ should have not sent '$prepare^3$' yet. In general, for all j, $1 \leq j \leq k$, if sites $(*, \ldots, *, Y_j, \ldots, Y_k)$ have not sent '$prepare^j$' then all sites $(*, \ldots, *, Y_{j+1}, \ldots, Y_k)$ should have not received all needed '$prepare^j$' and have not

sent '*prepare*$^{j+1}$'. Therefore, by induction, all sites $(*, \ldots, *, Y_k)$ have not sent '*prepare*^{k}' yet. So x should have not received '*prepare*^{k}' from site $(X_1, \ldots, X_{k-1}, Y_k)$. That means x will have not entered state *c* yet. It is contradictory to the assumption. Therefore, If a site is in state *c*, all sites have sent '*prepare*^{1}' messages. □

THEOREM 5.6 *Protocol 2 is a nonblocking protocol.*

PROOF:

1. By lemma 4.2, if a site is in commit state all other sites have to be in one of the p states. The abort state is not reachable from the p states. Therefore, no local state has both commit and abort state in its concurrency set.

2. The noncommitable states are state *q*, *w* states and *a* states. Again, by lemma 4.2, when a site is in one of these states no other site can be in commit state. Hence, there is no noncommitable state whose concurrency set contains commit state.

Therefore, the nonblocking conditions are satisfied. □

5.3 The Complexity of Message Exchange

Assume that N and k are such that $N^{1/k}$ is an integer. Since the number of messages sent in each round by a site is $(N^{1/k} - 1)$. The total number of messages needed by the blocking protocol is $kN(N^{1/k} - 1)$ and for nonblocking protocol is $2kN(N^{1/k} - 1)$. If $N^{1/k}$ is not an integer, we can add sufficient virtual sites to the system. The only difference between real sites and virtual sites is that virtual sites do not decide to abort the transaction by themself or the virtual sites have value 0 initially. The number of virtual sites needed is $(M - N)$, where $M^{1/k}$ is equal to $\lceil N^{1/k} \rceil$. Now the total number of messages needed would be $k\lceil N^{1/k} \rceil^k (\lceil N^{1/k} \rceil - 1)$ for blocking protocol and consensus protocols and $2k\lceil N^{1/k} \rceil^k (\lceil N^{1/k} \rceil - 1)$ for the nonblocking protocol. Therefore, the order of complexity are all $\Theta(kN^{1/k})$. If we ignore the ceiling operator and consider k to be real, the number of message is minimum for $k = \ln N$ and the corresponding message complexity is $\Theta(N \ln N)$

Obviously, these two families of commit protocols are optimal with respective to message. Because we have shown that for any decentralized commit protocol, which has k rounds of message interchanges, requires $\Theta(kNN^{1/k})$ messages. And the message lower bound is $\Theta(N \ln N)$ for any decentralized commit protocol.

6 Decentralized Consensus Protocols

The communication scheme used in section 5 can be simplified to apply on solving the following problems efficiently. Figure 5. is the simplified schema.

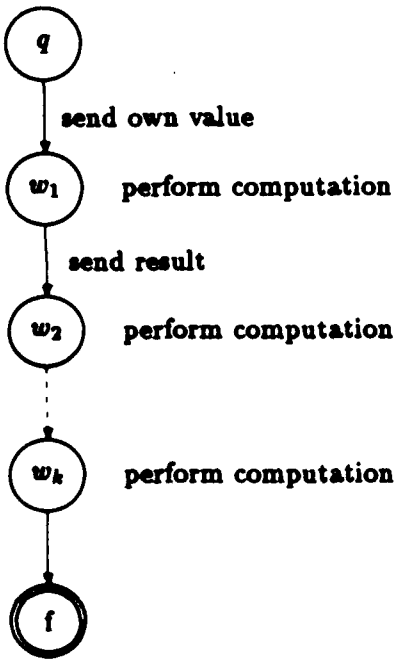


Figure 5: Consensus Protocol Schema

6.1 Decentralized Finding Maxima and Minima

Given N distinct values distributed across the sites of a distributed system, such that each site contains one value. The problem is to find the extreme (maximum or minimum) of these values and to make this extrema known to all sites in the system. The actions of each site is modeled as a FSA shown in figure 5. Assume $N^{1/k}$ is an integer then the actions in site x, which has binary representation $(x_1, \ldots, x_k)$, are as follows :

1. State q: Send own value to sites numbered as $(*, X_2, \ldots, X_k)$ and move to state w_1.

2. State w_i, $1 \leq i \leq (k-1)$: Upon receiving values from sites numbered as $(X_1, \ldots, X_{i-1}, *, X_{i+1}, \ldots, X_k)$, compute the extrema (maxima or minima) of these values received and the computed result of the previous state. Send the result to all sites numbered as $(X_1, \ldots, X_i, *, X_{i+2}, \ldots, X_k)$ and move to state w_{i+1}.

3. State w_k: Upon receiving values from all sites numbered as $(X_1, \ldots, X_{k-1}, *)$, compute the extrema (maxima or minima) of the values received and the computed result of previous state. Move to state f.

4. State f: Final state.

The following theorem establishes correctness of the protocol.

THEOREM 6.1 *If a site is in final state f then it contains the extrema (maxima or minima) of the values initially present.*

PROOF: Assume the extrema of the values initially present to be the value at site x, say V_x. Then all sites will contain V_x when they are in state f. We will show this by induction.

INDUCTIVE HYPOTHESIS: *After ith round of message exchange, where $1 \leq i \leq k$, all sites numbered as $(\underbrace{*, \ldots, *}_{i-bits}, X_{i+1}, \ldots, X_k)$ have received V_x.*

1. Base case: i is equal to 1: Site x has V_x in state q. It sends V_x to sites $(*, X_2, \ldots, X_k)$. Therefore, after the 1st round of message exchange, all sites $(*, X_2, \ldots, X_k)$ have the extrema V_x.

2. Inductive case: Assume that the hypothesis is true for $i = l-1$, where $l \leq k$: After the $(l-1)$th round of message exchange, all sites $(\underbrace{*, \ldots, *}_{(l-1)bits}, X_l, \ldots, X_k)$ have the extrema V_x. They will send V_x to all sites $(\underbrace{*, \ldots, *}_{l-bits}, X_{l+1}, \ldots, X_k)$. So after lth round of message exchange, all sites $(\underbrace{*, \ldots, *}_{l-bits}, X_{l+1}, \ldots, X_k)$ contain the extrema V_x and move to state f. Therefore, the inductive hypothesis is true for all i, where $1 \leq i \leq k$.

The only case a site is in state f is that it already goes through k rounds of message exchange, therefore it should contain the extrema. After kth round of message exchange

all sites $(\underbrace{*,\ldots,*}_{k-bits})$ contain extrema. So, If a site is in state f it contains the extrema of the values initially present. □

6.2 Computation of Sum Function

Given N values distributed across the sites of a distributed system, such that each site contains one value, the problem is to find the sum of these values and to make the sum known to all sites in the system. As before, the protocol in figure 5. can be used. We can again assume $N^{1/k}$ is integer and site number x can be representation as binary expansion $(X_1,\ldots,X_k)$. The actions in each state at a site x are explained below.

1. State q: Send own value to sites numbered as $(*,X_2,\ldots,X_k)$ and move to state w_1.
2. State w_i, $1 \leq i \leq (k-1)$: Upon receiving values from sites numbered as $(X_1,\ldots,X_{i-1},*,X_{i+1},\ldots,X_k)$, compute the sum of these values received and the computed result of the previous state. Send the result to all sites numbered as $(X_1,\ldots,X_i,*,X_{i+2},\ldots,X_k)$ and move to state w_{i+1}.
3. State w_k: Upon receiving values from all sites numbered as $(X_1,\ldots,X_{k-1},*)$, compute the sum of the values received and the computed result of the previous state. Move to state f.
4. State f: Final state.

The following theorem establishes correctness of the protocol.

THEOREM 6.2 *If a site is in state f then it contains the sum of the values initially present.*

PROOF: We will show it by induction. Assume any site x has value $V^0(x)$ initially.

INDUCTIVE HYPOTHESIS: *After ith round of message exchange, where $1 \leq i \leq k$ site x will have value $V^i(x)$. Where*

$$V^i(x) = \sum V^0(\underbrace{*,\ldots,*}_{i-bits},X_{i+1},\ldots,X_k) \tag{6}$$

1. **Base case:** When $i = 1$, after the 1st round of message exchange, site x has received values from sites
   $(*, X_2, \ldots, X_k)$. These sites have values $V^0(*, X_2, \ldots, X_k)$ initially, therefore, site x will contain sum of $V^0(*, X_2, \ldots, X_k)$.

2. **Inductive case:** Assume for $i = l - 1$, where $l \leq k$, the inductive hypothesis is true. After the $(l-1)$th round of message exchange, site x would have value $V^{l-1}(x)$ which is sum of $V^0(\underbrace{*, \ldots, *}_{(l-1)bits}, X_l, \ldots, X_k)$. After the lth round of message exchange site x would receive values from sites $(X_1, \ldots, X_{l-1}, *, X_{l+1}, \ldots, X_k)$ which have values $V^{l-1}(X_1, \ldots, X_{l-1}, *, X_{l+1}, \ldots, X_k)$, therefore $V^l(x)$ would be

$$\begin{aligned} V^l(x) &= \sum_{a=0}^{N^{1/k}-1} V^{l-1}(X_1, \ldots, X_{l-1}, a, X_{l+1}, \ldots, X_k) \\ &= \sum_{a=0}^{N^{1/k}-1} \sum V^0(\underbrace{*, \ldots, *}_{(l-1)bits}, a, X_{l+1}, \ldots, X_k) \\ &= \sum V^0(\underbrace{*, \ldots, *}_{l-bits}, X_{l+1}, \ldots, X_k) \end{aligned} \tag{7}$$

   So the inductive hypothesis is true for all i, $1 \leq i \leq k$.

The only case a site x is in state f is that it already goes through k rounds of message exchange, therefore it should contain the value $V^k(x)$. $V^k(x) = \sum V^0(\underbrace{*, \ldots, *}_{k-bits})$ is the sum of all values initially present, so if a site is in state f, it contains the sum of all values initially present. □

7 Conclusions

We have defined a communication structure for decentralized commit protocols which allows us to derive a family of decentralized commit protocols and also obtain a tradeoff between number of messages and the number of rounds of message exchange. The protocols are symmetric and need only $\Theta(kNN^{1/k})$ messages for k rounds of message interchanges. It has been shown that for any decentralized commit protocol that uses k rounds of message interchange needs $\Theta(kNN^{1/k})$ messages. Therefore, we have found a class of message optimal decentralized commit protocols. This communication structure can be used to derived decentralized consensus protocols with the same message complexity.

8 References

1. J. N. Gray, **Notes on Database Operating Systems**, in *Operating Systems: An Advanced Course*, Springer-Verlag, Berlin, 1979.

2. D. Skeen, **Nonblocking Commit Protocols**, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 133-142, 1981.

3. D. Skeen and M. Stonebraker, **A Formal Model of Crash Recovery in a Distributed System**, *IEEE Trans. on Software Eng.*, Vol. SE-9, pp. 219- 228, May 1983.

4. T.V. Lakshman and A. K. Agrawala, **Communication Structure of Decentralized Commit Protocols**, *Computer Science Technical Report TR-1489*, University of Maryland, College Park, Apr. 1985.

5. T.V. Lakshman and A.K. Agrawala, **Efficient Decentralized Consensus Protocols**, *IEEE trans. on Software Eng.*, Vol. SE-12, No. 5, pp. 600-607, May 1986.

END

9-87

DTIC